# INTEGRATION OF SELECTED COMPONENTS OF AN INTELLIGENT SYSTEM FOR EFFECTIVE ANALYSIS OF DIAGNOSTIC AND REPAIR WORK OF INDUSTRIAL DEVICES USING ADVANCED IMAGE ANALYSIS

**A Degree Thesis**

**Submitted to the Faculty of the**

**Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona**

**Universitat Politècnica de Catalunya**

**by**

**Oriol Griera Jiménez**

**In partial fulfilment**

**of the requirements for the degree in**

**TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING**

**Advisors: Mikołaj Leszczuk, Francisco Vallverdú**

**Barcelona and Krakow June 2020**

## Abstract

Nowadays, it is a fact that Artificial Intelligence is changing our lives completely. Advancements in machine learning and deep learning are creating a paradigm shift in virtually every sector of the tech industry. The goal of this project is to use these technologies to develop an innovative system which supports employees in the implementation of renovation, repair, and diagnostic work on industrial equipment.

The result has been an application implemented with Python, which can detect and classify four different industrial objects from images. These multimedia files are obtained from video records of the activities performed in the factory. Finally, it is needed to say that this classifier is a module of a bigger system which, among others, contains more classifiers.

## Resum

Avui en dia, és un fet que la Intel·ligència Artificial està transformant per complet les nostres vides. Els progressos en “machine learning” i “deep learning” estan provocant un canvi de paradigma en els sectors de la indústria tecnològica. L’objectiu d’aquest projecte és utilitzar aquestes tecnologies per desenvolupar un sistema innovador que ajudi als treballadors encarregats de renovar, reparar, i diagnosticar errors en l’equipament industrial.

El resultat ha estat una aplicació implementada amb Python, capaç de detectar i classificar quatre objectes industrials diferents a partir d’imatges. Els fitxers multimèdia s’obtenen de gravacions de vídeo de les activitats que es desenvolupen a la fàbrica. Finalment, cal mencionar que aquesta aplicació és una part d’un sistema més gran que, entre altres, conté més mòduls classificadors.

# Resumen

Hoy en día, es un hecho que la Inteligencia Artificial está transformando por completo nuestras vidas. Los progresos en “machine learning” y “deep learning” están causando un cambio de paradigma en los sectores de la industria tecnológica. El objetivo de este proyecto es usar estas tecnologías para desarrollar un sistema innovador que ayude a los trabajadores encargados de renovar, reparar, y diagnosticar errores en el equipamiento industrial.

El resultado es una aplicación desarrollada con Python, capaz de detectar y clasificar cuatro objetos industriales diferentes a partir de imágenes. Los ficheros multimedia se obtienen de grabaciones de vídeo de las actividades que se desarrollan en la fábrica. Finalmente, hay que mencionar que esta aplicación es una parte de un sistema más grande que, entre otros, contiene más módulos clasificadores.

## Acknowledgements

I would like to express my gratitude to all the INRED members for allowing me to participate in this project. I want to thank Professor Andrzej Zeja and Konrad for their help during the project, and especially to my supervisor Professor Mikołaj Leszczuk, for all his support during these months and given the difficult situation caused by the pandemic, making my work as more comfortable as possible.

# Revision history and approval record

| Revision | Date | Purpose |
|---|---|---|
| 0 | 10/05/2020 | Document creation |
| 1 | 26/06/2020 | Document revision |
| | | |
| | | |
| | | |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Oriol Griera Jiménez | oriolgrierajimenez@gmail.com |
| Mikołaj Leszczuk | mikolaj@leszcz.uk |
| Francisco Vallverdú | sisco.vallverdu@upc.edu |
| | |
| | |
| | |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Date | 12/06/2020 | Date | 26/06/2020 |
| Name | Oriol Griera | Name | Mikołaj Leszczuk, Francisco Vallverdú |
| Position | Project Author | Position | Project Supervisor |

# Table of contents

# List of Figures

# List of Tables:

# 1. Introduction

The trend to the automation is being accelerated by daily advances in Artificial Intelligence and Deep Learning. In the industry, the automation of tasks by machines has become one of the biggest technological revolutions in this field.

A crucial part of automation is Computer Vision. It allows computers to “see” as humans do through the acquisition, processing, and analysis of digital images and videos. To do so, it uses a trained algorithm which identifies specific details within an image.

The benefits generated by these technologies in the industry is enormous. It allows us to perform tasks faster and more accurately, reducing human error. And the best is yet to come; we are only at the beginning with a whole world to explore

## 1.1. Statement of purpose

The purpose of this project is to develop selected components of a system which ensures the integration of complex subsystems of intelligent monitoring, quality control of renovation and repair works as well as fault diagnosis, with simultaneous comprehensive processing and exchange of information between system modules. The proposed system includes solutions that can be classified into two groups of issues:

- intelligent diagnostics system
- information processing and exchange system

The module developed in this project is an application that detects and classifies industrial objects from multimedia files.

## 1.2. Requirements and specifications

Talking about the project requirements, it is necessary to ensure a minimum of accuracy in the classification of different objects. Therefore, objects must be detected with at least a certainty of 60%.

Moreover, as this module is part of a bigger system, it is essential to guarantee the correct integration with the rest of the modules. Consequently, the output data generated by the application is not only visual but also text in JSON format.

As a specification, the database to train and test the algorithm is obtained from multimedia files provided by INRED Research Group. There are seven videos with a total of 3,5 hours of activities recorded in a factory.

## 1.3. Methods and procedures

This project is part of a larger project called INRED. INRED is a research project developed by the AGH University of Science and Technology with an external company at the Faculty of Computer Science, Electronics and Telecommunications in Krakow. Its main goal is developing a system to improve the industry sector by facilitating the collection of data and its subsequent analysis to determinate further improvements in the implementation of industrial activities.

As INRED started in 2018, the first step in this project was collecting all the code previously done by another student and integrate it to make it run as a system.

The application has been developed with Python. It uses the TensorFlow Object Detection API and Flask to implement the RESTful API. The User Interface is done with Flasgger. Also, it has been applied an open-source software called LabelImg to label the pictures.

## 1.4. Work Plan

This section contains the three Work Packages in which this project is divided, the milestones and the Gantt diagram.

### 1.4.1. Work Packages

<table>
<tr><td colspan="2">Project: Work environment preparation</td><td colspan="2">WP ref: (WP1)</td></tr>
<tr><td colspan="2">Major constituent: Software</td><td colspan="2">Sheet 1 of 1</td></tr>
<tr><td colspan="2" rowspan="2">Short description:<br>It is necessary to collect all the different code previously done in the virtual machine.</td><td colspan="2">Planned start date: 08/03/2020<br>Planned end date: 06/04/2020</td></tr>
<tr><td colspan="2">Start event: 08/03/2020<br>End event: 06/04/2020</td></tr>
<tr><td>Internal task T1: Installation of the virtual machine.<br>Internal task T2: Accumulate the different modules in the virtual machine and learn how to run them.</td><td>Deliverables:<br>None</td><td>Dates:<br>None</td></tr>
</table>

***Table 1:*** *Work Package 1 - Work environment preparation*

| | | |
|---|---|---|
| Project: System development | WP ref: (WP2) | |
| Major constituent: Software | Sheet 1 of 1 | |
| Short description:<br>Codification of different scripts in Python to ensure the work of all the different modules as a system to train the model. Implementation of the User Interface. | Planned start date: 07/04/2020<br>Planned end date: 22/05/2020 | |
| | Start event: 07/04/2020<br>End event: 22/05/2020 | |
| Internal task T1: Development of the scripts | Deliverables:<br>- Provisional scripts | Dates:<br>14/05/2020 |

***Table 2:*** *Work Package 2 - System development*

| | | |
|---|---|---|
| Project: Testing | WP ref: (WP3) | |
| Major constituent: Simulation and Software | Sheet 1 of 1 | |
| Short description:<br>This part corresponds to the testing of the system and the resolution of defects. | Planned start date: 23/05/2020<br>Planned end date: 12/06/2020 | |
| | Start event: 23/05/2020<br>End event: 12/06/2020 | |
| Internal task T1: System testing<br>Internal task T2: Resolution of defects<br>Internal task T3: Project documentation | Deliverables:<br>- Final scripts<br>- Evidences<br>- User Manual<br>- Project documentation | Dates:<br>12/06/2020 |

***Table 3:*** *Work Package 3 - Testing*

### 1.4.2. Milestones

| WP# | Task# | Short title | Milestone / deliverable | Week |
|---|---|---|---|---|
| 1 | 1 | Installation of the virtual machine | None | - |
| 1 | 2 | Accumulate the different modules in the virtual machine and learn how to run them. | None | - |
| 2 | 1 | Development of the scripts | Provisional Scripts | 11 |
| 3 | 1 | System testing | None | 13 |
| 3 | 2 | Resolution of defects | Final scripts and evidences | 13 |
| 3 | 3 | Project documentation | User Manual and Project documentation | 13 |

***Table 4:*** *Milestones*

### 1.4.3. Gantt diagram

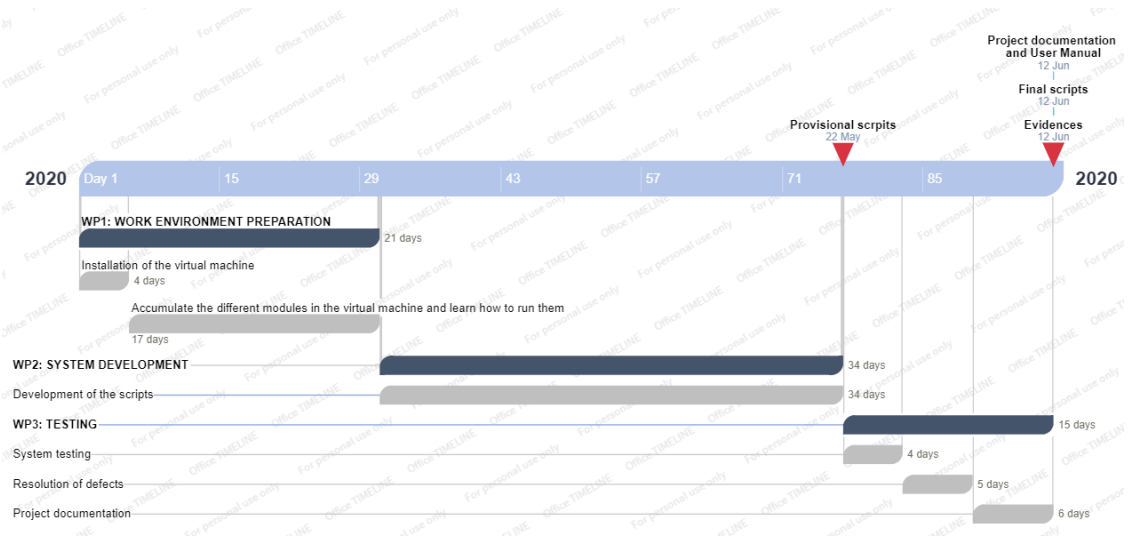



***Figure 1:*** *Gantt diagram*

## 1.5. Deviations and Incidences

The development of the project has been affected by the actual situation regarding the COVID-19 virus. Due to the quarantine, Polish universities have been closed during the stay, and the communication with the supervisor has been via online. As a result, it took more time to start developing the project.

The main deviation from the initial plan is the duration of WP1 “Work environment preparation”. It had to be modified because of the necessity to move the virtual machine with the work environment from VirtualBox to Proxmox. The leading cause is the lack of capacity of the computer device used, which caused difficulties when processing the training of the model. To solve the incidence, an external server has been used for computation.

# 2. State of the art of the technology used or applied in this thesis:

In this section, it is explained a resume of the research done on object detection during the last few years.

## 2.1. Object detection

Object detection has recently raised as one of the most exciting subjects in Computer Vision and AI fields. The possibility to perform real-time recognition of objects in a scene has opened a whole world of possibilities. The first appearance was in 2012 when AlexNet developed CNN and won the ILSVRC competition [1]. Since then, new models to perform detections have been grown, such as R-CNN, Fast R-CNN and Faster R-CNN.

### 2.1.1. Convolutional Neural Networks

CNN are neural networks that use convolution, usually for analysing images. The primary purpose of convolution is to extract features from input images preserving the spatial relationship between pixels [2]. To do so, it learns image features using small squares of input data.

The structure of a CNN is based in a sequence of layers. The three main types of layers are Convolutional layer, Pooling layer and Fully-connected layer [3]. In Figure 2, it can be seen an example of a CNN. As it is shown, CNN can combine a different number of Convolutional and Pooling layers.

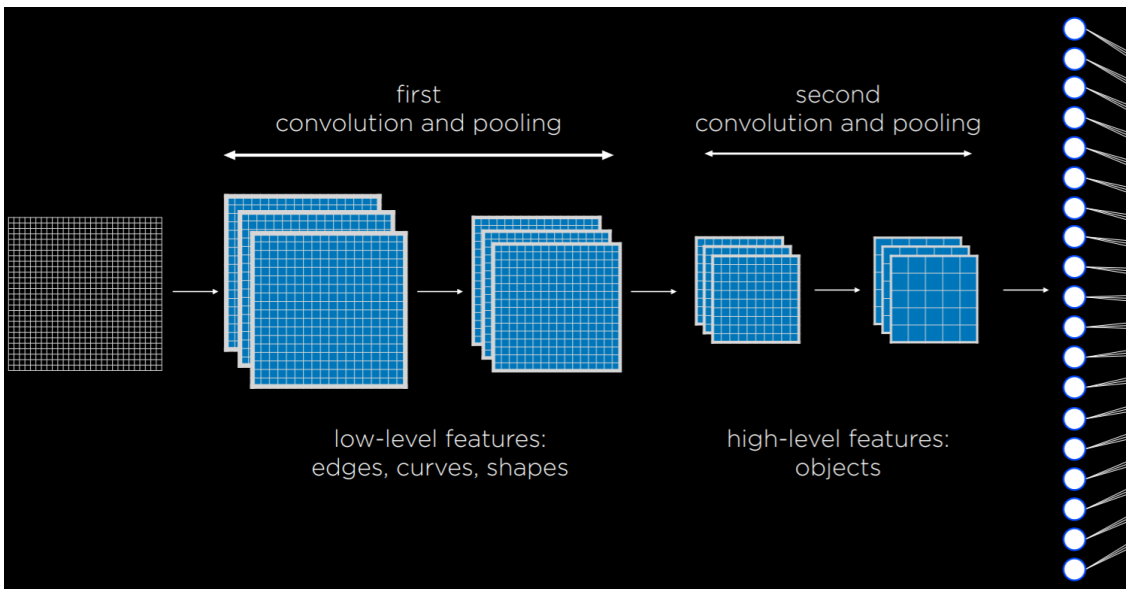



***Figure 2:*** *CNN architecture*

The Convolutional layer is responsible for all the calculations needed to extract information from images (feature maps). By applying a different type of filters, features such as edges, curves, and shapes can be identified [4]. To create the layer, it is required to introduce parameters like the number of filters or kernels and their dimensions, the number of input channels and output channels, and the depth of the convolution filter.

The Pooling layer is used to reduce the size of input by combining outputs of neuron clusters at one layer into a single neuron in the next layer. The result is calculated either by choosing the maximum value (max pooling) or using the average of the values (average pooling).

The process of flattening consists of converting the features map into a vector. Then, each neuron is connected to different Fully-connected layers (also known as Dense layers) to generate the prediction.

#### 2.1.1.1. R-CNN

Region Convolutional Neural Networks are CNN based on a selective search which consists in dividing the original image into a fixed number of region proposals. The process of detection using this type of neural network is illustrated in Figure 3 [5].

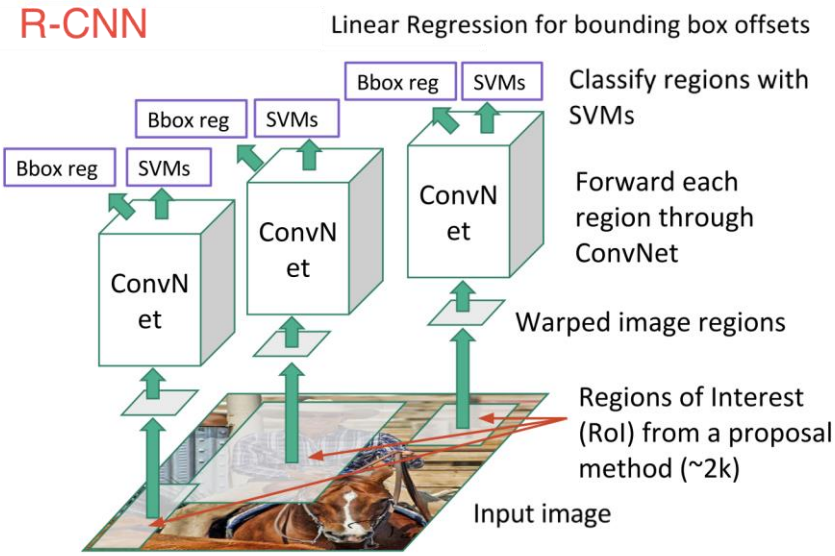



***Figure 3:*** *R-CNN*

The idea of this process is turning the object detection into an image classification problem. First, the input image is scanned using the selective search algorithm, generating 2000 region proposals. Then, it is run locally in each region a CNN [6]. On the one hand, the output of the CNNs is used to feed the SVM for classifying the area, and on the other hand, in case the object exists, as an input to a linear regressor in charge of tightening the bounding box of the object.

The main problem of this type of CNN is that it requires a significant amount of time to train the model. What is more, the algorithm cannot be implemented in real-time as it takes a long period to perform the detection.

### 2.1.1.2. Fast R-CNN

Fast R-CNN was developed to improve the detection speed of its processor, R-CNN. Mainly, the most significant changes are generating the feature map over the image before creating the region proposals and replacing the SVM with a SoftMax layer [7].

With these improvements, the training and testing time is considerably reduced. Instead of repeating the process of convolution for the 2000 regions, the convolution operation is done once.

In Figure 4, it can be seen the Fast R-CNN architecture [5].

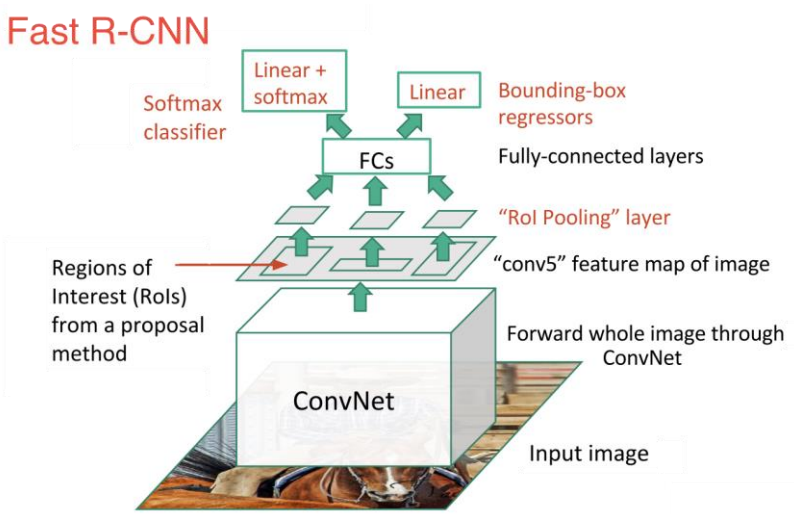



***Figure 4:*** *Fast R-CNN*

However, even though it brought improvements, it was necessary to modify the selective search algorithm for generating the region proposals.

### 2.1.1.3. Faster R-CNN

Both Convolution Neural Networks explained before use the selective search algorithm to find out the regions. This process is slow and time-consuming, which affects the performance of the model [8]. In 2015, Shaoquing Ren et al. developed an object detection algorithm with which the selective search was not required. Instead, the network learns the region proposals.

The main difference is that is used a separate Region Proposal Network (RPN) to predict the regions. Once done, the predictions are made; they are filtered with an RoI pooling kernel and used to feed a Fast R-CNN [9]. The results are the name of the predicted object and its bounding boxes.

As a result, Faster R-CNN is the fastest CNN and can be used to perform detections in real-time.

# 3. Project development:

This part is where the core work carried out during the project is explained. The job done is divided into three groups: the integration of the pieces of code used for the extraction and labelling of images from the dataset to prepare the training and testing of the algorithm, the training and extraction of a model able to detect and classify objects, and the integration of the model with the user interface to display the results.

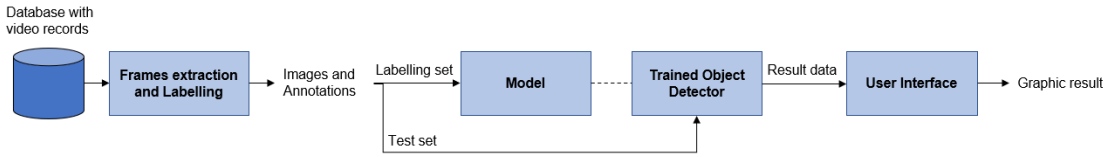



***Figure 5:*** *System operation diagram*

The diagram in Figure 5 shows schematically how the system works. The system development is divided into two parts: the process to obtain the trained model, and the usage of this model in the application. In both pieces, it is used the frames extracted from the video records: on the one hand, to train and test the model and, on the other hand, as input data to detect the objects. In the next points, both parts are explained in detail.

## 3.1. Preparation of the data

Since the input of the detector needs to be an image, the first step is obtaining the frames from the videos with the objects willing to detect. Afterwards, to train the model, it is necessary to label the images and create annotations. These files are the source where the algorithm takes the knowledge from.

### 3.1.1. Data set

The INRED project database contains a large number of video records obtained from filming activities developed in the industry. For each of the events, it is necessary to recognise the key objects that appear in the videos, as well as the machines used to transport the objects.

In this project, the application has been developed to detect four different objects: tensioning spring, pressure spring, metal sheet, and balls. In Figure 6, it is illustrated as an example of each object.

***Figure 6:*** *Ball, Metal sheet, Pressure spring, Tensioning spring*

These objects are present in seven records which means approximately 3,5 hours of video. To extract the images, each video is split up into frames using a Python program. Considering that a frame is saved at a rate of 1fps, this means approximately 12600 images to work with. Dealing with a such number of pictures requires spending a lot of time labelling and a great computational effort.

To reduce the amount of data, it is only considered the parts of the video where movement is detected. To do so, pictures with a gap of 2s are compared to extract some differences. If these differences exceed 5000px, then it is considered as movement, and if the motion is detected during 20s or more, it is enough to create a new video. The fact of training the model with images in which obtaining a neat picture of the object is more complicated and therefore detecting it, increases the classifier efficiency when performing identifications with static images.

The reduced data set contains a total number of 1624 frames, which are divided into two groups: labelling set (60%) and test set (25%). The remaining pictures are not used as they do not have the objects willing to detect.

The first set consists of the photograms which, after the process of labelling explained in the next point, will be used to generate the knowledge of the model. The test set is used to check the object detection with images that the algorithm has never seen before.

### 3.1.2. Labelling

The process of labelling consists in, for each frame, creating an XML file with the name of the objects showed and the coordinates (in pixels) of their positions. LabelImg is an open-source software which facilitates the task by allowing to realise this process graphically. In Figure 7, it can be seen a screenshot of the program and an example of the XML file created.

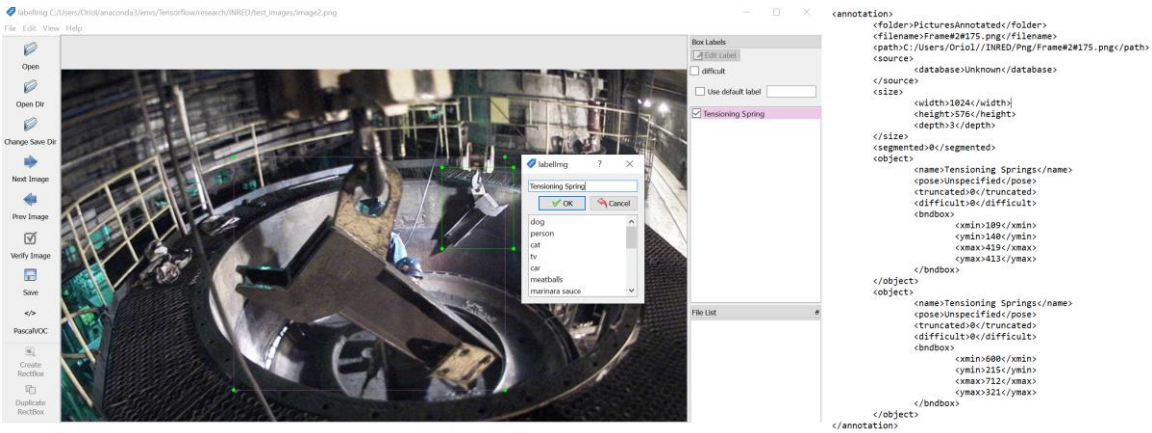



***Figure 7:*** *Process of labelling and an XML file example*

The number of pictures manually annotated is 979. Considering that the images are taken by cameras and can be placed in different places, it is interesting to have different points of view for recognition. The results are improved by using symmetry on pictures simulating two different points of view. Annotations of the new frames created by mirroring are automatically done using a Python program [10]. As a result, the labelling set is increased up to 1958 annotated frames.

## 3.2. Object Detector

The model used as the classifier is a Convolutional Neural Network extracted from the TensorFlow deep learning framework. After the study of the characteristics of different algorithms, the model used is Faster RCNN with Inception Resnet v2 [11]. Due to its features, it performs real-time detections with high accuracy. Moreover, it is implemented with TensorFlow Object Detection API [12], which is easy to work with and intuitive.

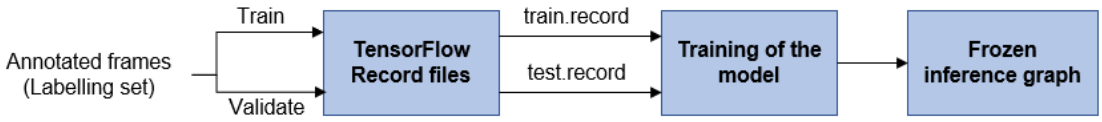



***Figure 8:*** *Process of the obtention of the trained model*

Regarding the acquirement of the trained model, Figure 8 shows the process in which using the labelled pictures, record files are created, and then the neural network is trained. The result is the inference graph. The frozen inference graph is the file where the exported model is saved and used to detect the objects in the application.

### 3.2.1. Training

The first step to perform the training of the model is to separate the annotated frames in two groups: 70% as the train set (1373 frames) and 30% as the validation set (585 frames).

The train set is the set of data used to train the model. During each epoch, the model is trained repeatedly on the same data in our training set, and it will continue to learn about the features of this data [13]. The hope with this is that later, the model can be deployed, and it can accurately predict new data that has never seen before based on its knowledge.

The validation set is a set of data separate from the training set that is used to validate the model during training. This validation process helps by giving information that may avoid the fact of overfitting. Overfitting happens when a model fits too closely to a particular dataset and therefore may fail to generalise to future data.

During the training process, the model classifies the output for each input in the training set. After this classification occurs, the loss is calculated, and the weights in the model are adjusted. Then, during the next iteration, the same input data will be classified.

Meanwhile, during each iteration of the training, the model classifies the inputs from the validation set based on the knowledge learnt. The difference with the train set is that after calculating the loss, the weights are not modified.

The utility of this process can be seen when we compare the results from classifying the two sets. If positive results are obtained from both sets, it is more probable that the model is not overfitting.

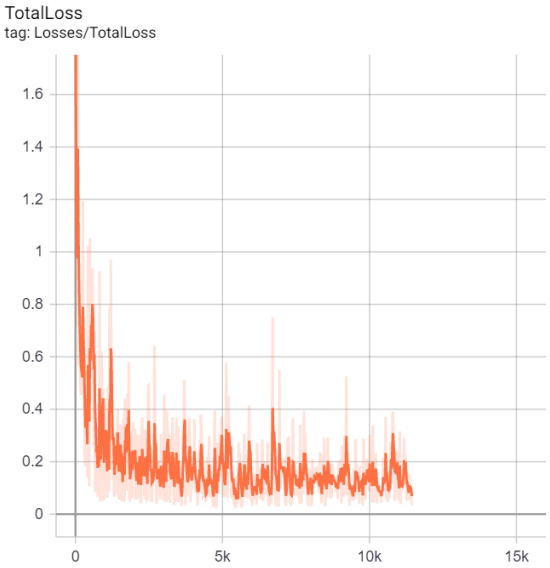



***Figure 9:*** *Total loss evolution*

To evaluate the training, TensorBoard API is used to visualise some metrics. In Figure 9, it is showing the evolution of the total loss during the training process. After 11613 iterations, the total loss is 0,089. The loss function determines how far the predicted values deviate from the actual values in the training data.

### 3.2.2. Result data

In this point, it is explained the format used to give the result data from the classifier. As mentioned before, this detector is a part of a more extensive system which has other modules that are going to use the information provided by the application. In consequence, it is required to think a particular format and structure of the result data which every module can process and understand as fast as possible. In this case, the result is given both in a graphic format and text format.

The TensorFlow Object Detection API provides the output data of the detector as a Python dictionary. The parameters saved are the number of detections, the name of the recognised objects, the coordinates where the objects are (expressed in image pixels), and the certainty of the discovery.

Certainty is a number between 0 and 1 that indicates confidence that the object was genuinely detected. The closer the number is to 1, the more confident the model is. It has to be mentioned that only objects recognised with a certainty equal or higher than 0.6 are included in the results.

As said before, one of the methods to represent the data is as an image. By giving the result graphically, it is easier for the human eye to process the information and evaluate whether the detection has been successful or not. It is showed in Figure 10 an example of the discovery of the object Tensioning Spring. As it can be seen, the result is a square box which surrounds the detected object, with the name of the object and the certainty of the detection.

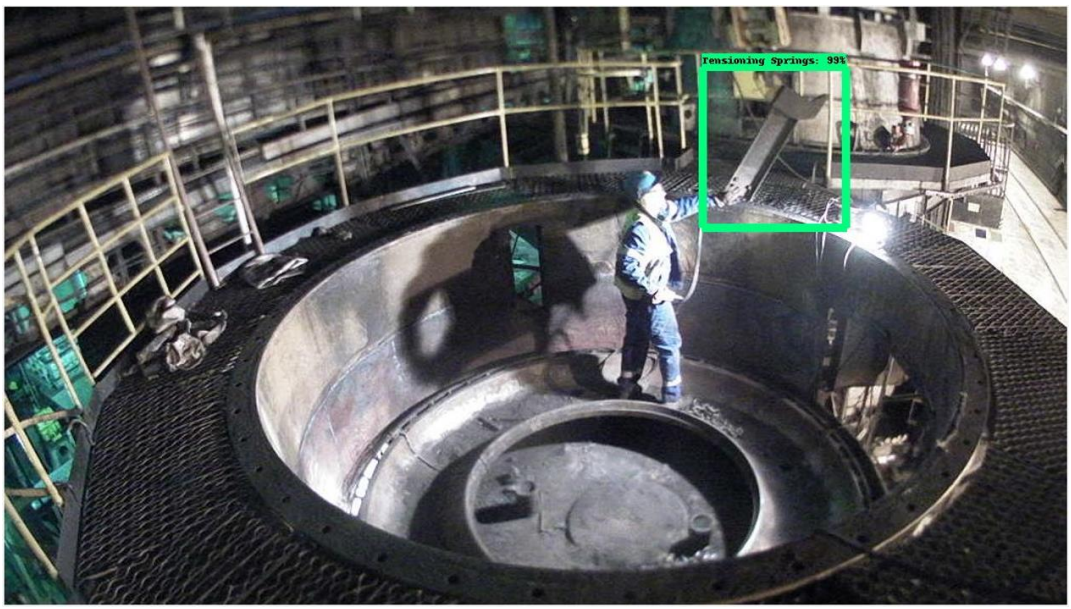



***Figure 10:*** *Graphic result*

However, contrary to humans, it is easier for a machine to process text format results. For that reason, the same information can be returned in JSON format. To do so, two Python classes are created: Ranking class and Recognition class. In Figure 11, there is a class diagram representing the relation between the two classes.

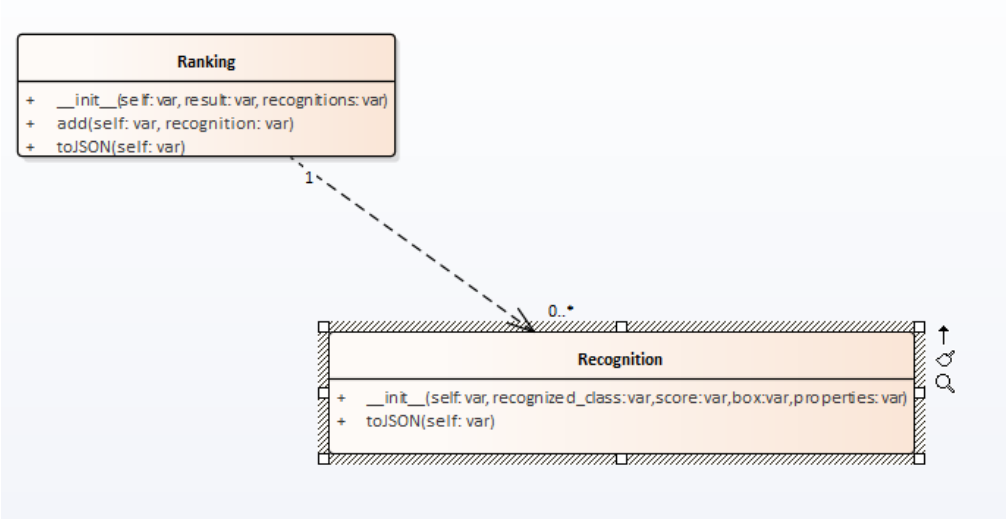



***Figure 11:*** *Ranking and Recognition class diagram*

The final result is given by applying the toJSON method to the Ranking object. A Ranking object has as an attribute a set of Recognition objects which are added by the add method. For each detection, a Recognition object is created with the information given by the model. In Figure 12, it is showed the same results as Figure 10, but serialised to JSON format.


```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Tensioning Springs",
            "certainty": "0.99164283",
            "box": [
                378,
                105,
                456,
                378
            ],
            "properties": {}
        }
    ]
}
```


***Figure 12:*** *Text result*

### 3.3. User Interface

In this point, it is explained the graphic interface of the application and its interaction with the object detector. The interface is used to select the images which are going to be used and to show the results obtained after the classifier processes the data.

#### 3.3.1. RESTful API

As the INRED system is based on SOA oriented architecture, the application provides RESTful API service. For that reason, the main code uses a web service template developed on the Python framework Flask.

A RESTful API is based on representational state transfer (REST), an architectural style and approach to communications used in web services development [14]. REST is accessible due to its simplicity and the fact that it builds upon existing systems and features of the internet's Hypertext Transfer Protocol (HTTP) defined by the RFC 2616 protocol [15]. Furthermore, REST technology is generally preferred over other technologies like SOAP because it uses less bandwidth, making it more suitable for efficient internet usage. Moreover, REST allows users to connect to, manage, and interact with cloud services. This feature is needed to access the database where all the videos are placed.

The diagram in Figure 13 represents the communication between the different parts of the application.

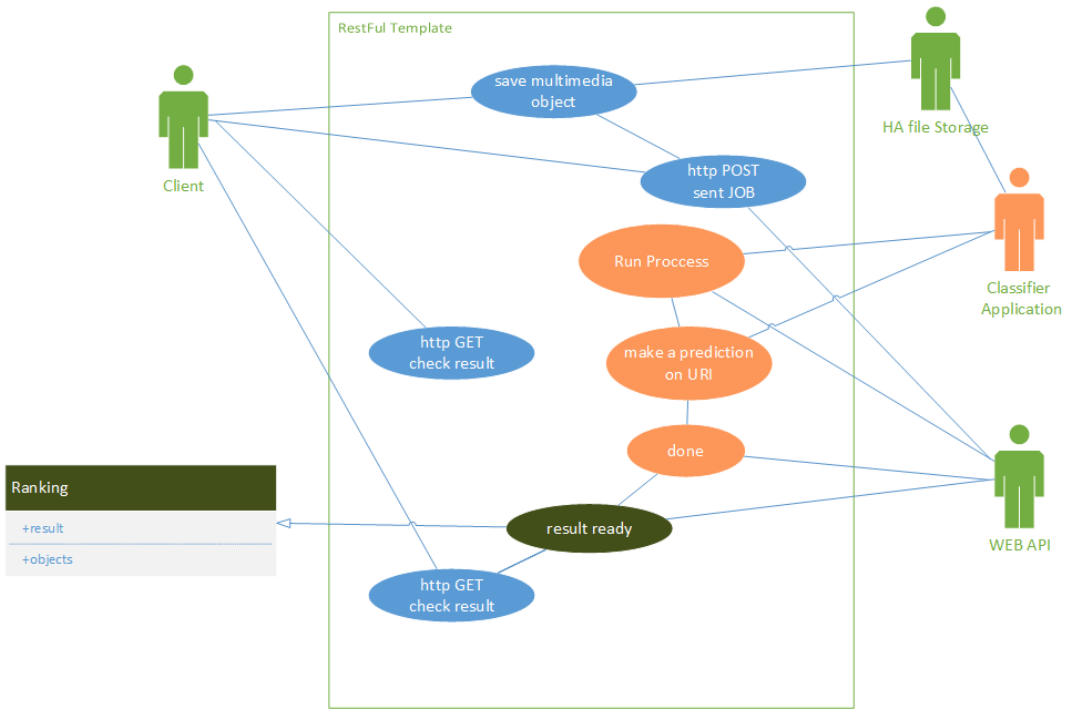



***Figure 13:*** *Connection diagram of the application elements*

The client orders to detect the objects of a particular image by the POST method. Then, a thread starts working and sends the task to the classifier, which has access to the High Availability storage that stores all the files. After the prediction is made, the client checks the result with a GET method.

### 3.3.2. Webpage templates

The interface has been developed with Flasgger. Flasgger is a Flask extension which facilitates the development of an API by extracting OpenAPI-Specification from all Flask views registered in the API created [16].

The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

What is more, Flasgger also comes with SwaggerUI embedded, which allows anyone to visualise and interact with the API's resources without having any of the implementation logic in place.

To explain the different templates used in the application, we can divide the process of using the app in three steps temporally: sending the order to detect, while the task is running, and when the result is obtained. All three levels can be seen in Figure 14.

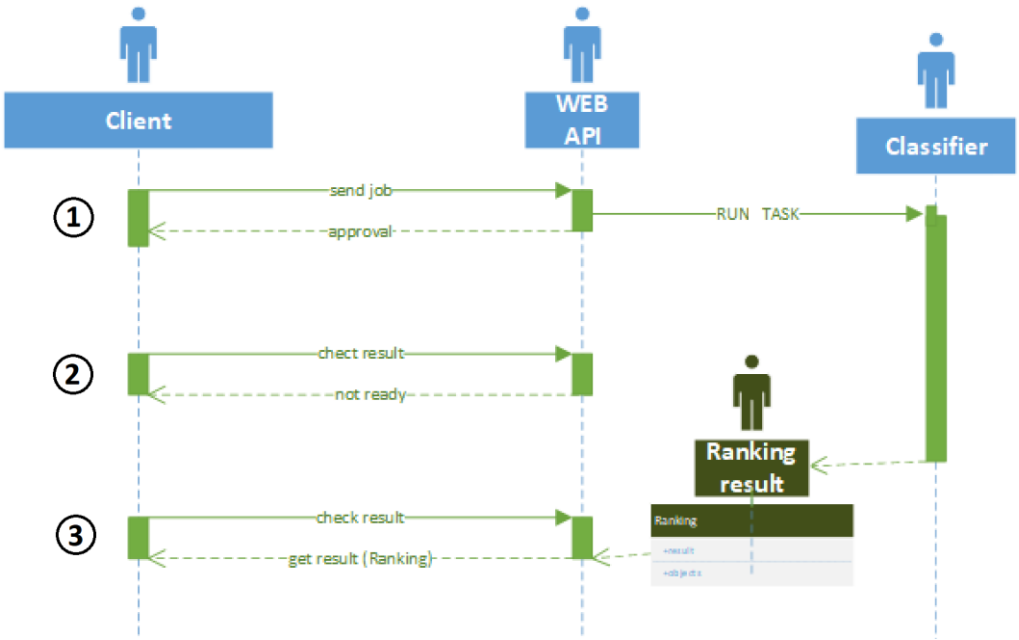



***Figure 14:*** *Application usage process*

The first step consists of sending the image in which the detection is going to be done. The parameters needed are the image and the Job ID. The Job ID is used to differentiate the threads and is provided in the *Information's* section. The template of this step is shown in Figure 15. The *Detect* button is used to initialise the detection (POST method), the *Get status* button is used to see the results (GET method), and there is also the *Web API* button, which redirects to a webpage that contains API documentation.

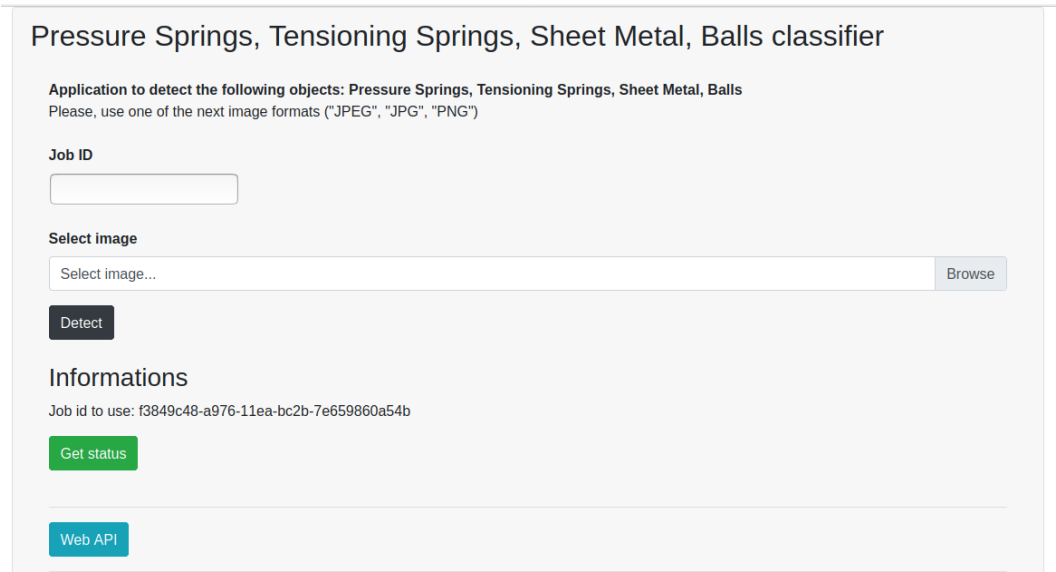



***Figure 15:*** *Web template 1*

The second template is used when the client tries to obtain the result before the detection has been done. In this case, the client is redirected to the model shown in Figure 16, which contains a message that says that the discovery is being processed.

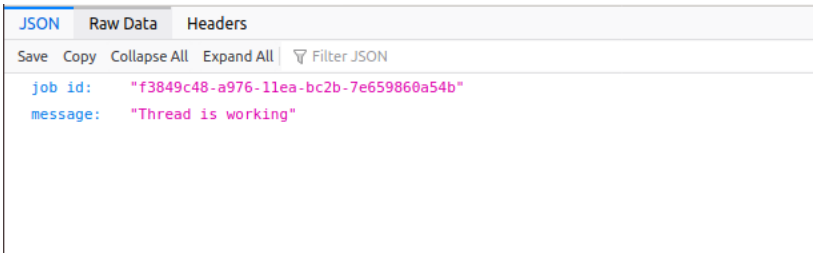



***Figure 16:*** *Web template 2*

Finally, the third template is used when the result is ready, and the client clicks on the *Get Status* button. It collects the information in the Ranking object created by the model, and it shows the result, both in graphic and text format. Also, it has the *Home* button to return to the first template. An example of the model can be seen in Figure 17.

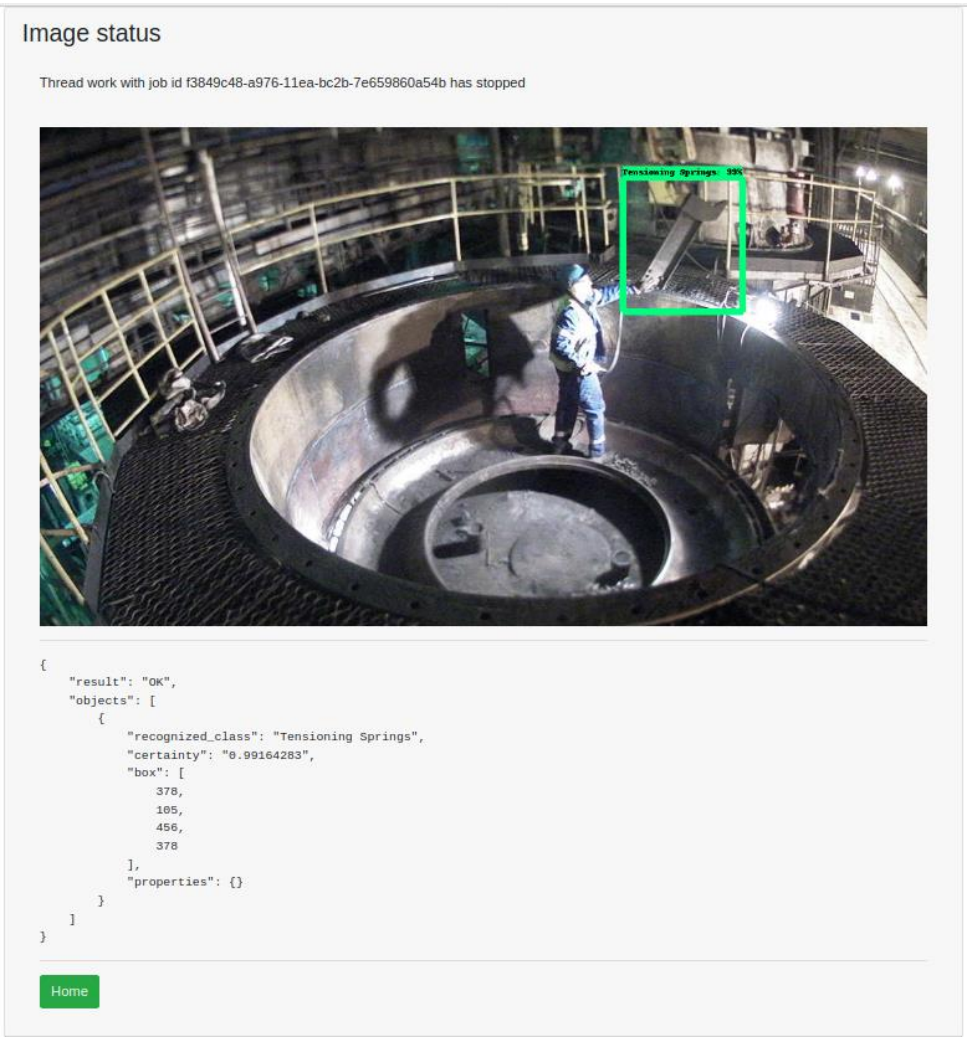



***Figure 17:*** *Web template 3*

# 4. Results

To analyse the performance of the object detector, the validation set has been used. This set of images consist of 400 frames from the video records: half of them are images which have the objects (50 per object), and the other half are negative examples with no artefacts.

When using the detector with the image, it can result in four different cases:

- **True Positive (TP)**: Detecting the correct objects in an image with objects (Figure 18).

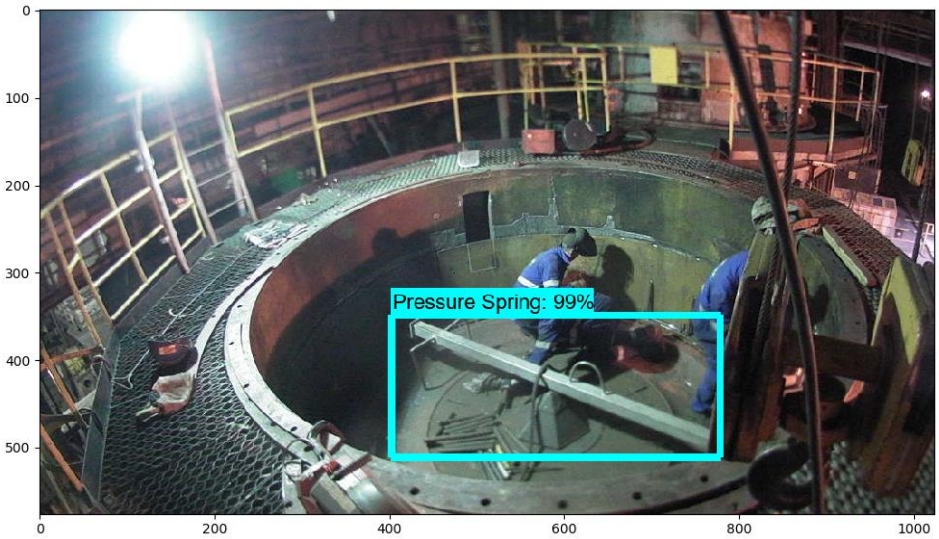



***Figure 18:*** *True Positive example*

- **False Positive (FP)**: Detecting an object in an image without artefacts (Figure 19).

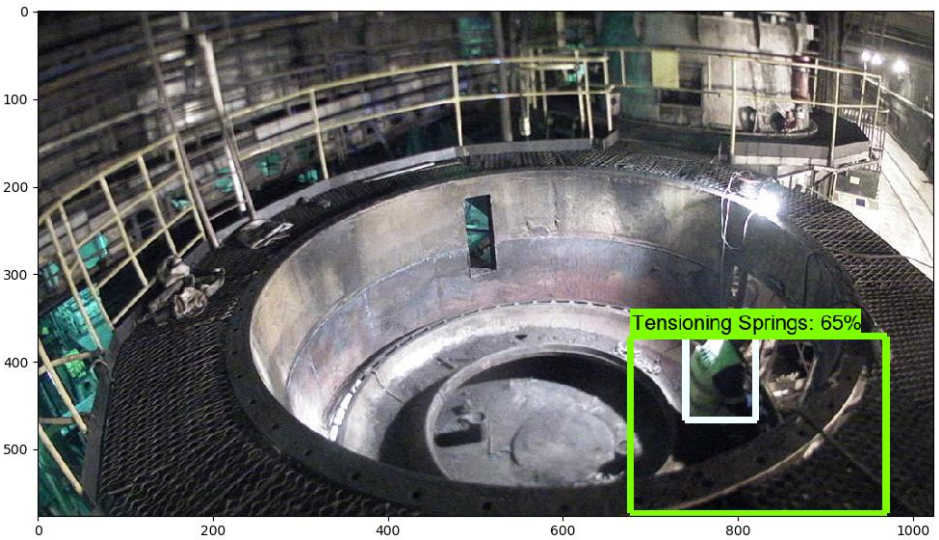



***Figure 19:*** *False Positive example*

- **True Negative (TN)**: No detection of objects in an image without objects (Figure 20).


***Figure 20:*** *True Negative example*

- **False Negative (FN)**: No detection of objects in an image with objects (Figure 21).


***Figure 21:*** *False Negative example*

After testing with the 400 images, the results are shown in Table 5.

| True Positives | False Positives | True Negatives | False Negatives |
|---|---|---|---|
| 193 | 15 | 185 | 7 |

***Table 5:*** *TP, FP, TN, and FP*

Additionally, true positives and false negatives can be counted for each type of object (Table 6).

| Object | True Positives | False Negatives |
|---|---|---|
| Tensioning Spring | 47 | 3 |
| Pressure Spring | 47 | 3 |
| Metal Sheet | 49 | 1 |
| Balls | 50 | 0 |

***Table 6:*** *TP and FN for each object*

The evaluation metrics used to measure the quality of the model are precision, recall and accuracy. Precision is the proportion of correctly detected objects from the total number of detections. The recall is the fraction of the frames with objects that are successfully detected. In other words, precision is how many detected items are relevant, and recall is how many related issues are identified. The third metric, accuracy, is the proportion of correct predictions among the total number of samples.

The definition of the metrics is defined in their equations:

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN} \qquad Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Finally, Table 7 shows the results after testing the model:

| Precision | Recall | Accuracy |
|---|---|---|
| 0.93 | 0.965 | 0.945 |

***Table 7:*** *Evaluation metrics*

In terms of precision, 93% of the detected objects are classified correctly. It means that the model can differentiate with a reasonable probability each type of object, and it does not fail into misleadingly identifying the background items.

Talking about a recall, the algorithm can detect rightly the 96.5% of the objects that appear in the images. This can be seen as the proportion of false negatives compared to true positives is significantly lower.

Looking at the results in terms of accuracy, the model is capable of classifying the objects when they are present in the images, as well as no detecting them when it is not necessary.

Also, it is useful to calculate the recall for each object. With this information, it is easy to control whether the detection of a specific object is failing, and more training frames are needed. Table 8 illustrates the results.

| | Tensioning Spring | Pressure Spring | Metal Sheet | Balls |
|---|---|---|---|---|
| Recall | 0.94 | 0.94 | 0.98 | 1 |

***Table 8:*** *Recall for each object*

As can be seen, the model can detect balls in all the images in which are present. Furthermore, the results with the other objects are considered positive as the minimum value has been 94%.

# 5. Budget

The costs of this project are divided into human resources and the hardware needed. As all the programs used are open source, no fees are related to software. To calculate the budget, it has been considered a project duration of 540 hours (equivalent to 18 ECTS).

On the one hand, the human resources cost includes the gross salary of a junior employee and social charges. A junior employee is paid 9€/h worked, which means a total wage of 4860€. The standard fees are 33% of the gross salary, which represents a cost of 1603.8€.

On the other hand, the hardware needed to develop the project is a computer with enough capacity to run the virtual machine. To calculate the amortisation, it is considered a computer price of 600€, a residual value of 60€, and a life span of 5 years. As a result, the first year's amortisation cost is 108€.

| Concept | Cost (€) |
|---|---|
| Gross salary | 4860 |
| Social charges (33%) | 1603.8 |
| Computer amortization | 108 |
| **TOTAL** | **6571.8** |

***Table 9:*** *Project costs*

As can be seen in Table 9, the total cost of the project has been of 6571.8€.

# 6. Conclusions and future development:

This project started with the objective of developing an application which has the capacity of detecting and classifying industrial objects from images. After extensive research during these months, it can be said that the goal has been achieved. The detector can distinguish with success four objects: the tensioning spring, the pressure spring, the metal sheet and the balls.

Moreover, the module has been successfully integrated into the INRED system. As the application was a part of a more extensive network, it was necessary to ensure the correct link-up between the modules. To do so, it has been figured out the way to express the detection results not only graphically but as text.

After realising the test to measure the quality of the module's performance, it can be concluded that the classifier can differentiate clearly whether an image contains one of the objects or not, and in such a case, give its location accurately. Even though the results mentioned may vary if more samples are evaluated, 400 images provide a reasonable idea of how the model is working.

Regarding the potential improvements, it could be interesting to apply more image processing techniques to augment the training data. In this project, it has been used mirroring to duplicate the number of samples to train the model, which simulates recording the objects from two different points of view. However, other affine transformations like rotating or shearing can be used.

Furthermore, it has been mentioned during the project but not implemented to avoid the effects of overfitting; it is useful to train the model with negative examples. By showing the model examples of images without objects, its knowledge is improved, and the probability of detecting false positives is considerably reduced.

Finally, mention that the technologies used in this project do not only have a successful present but a promising future. Though this project has an impact on the industrial sector, this application could be used in any field were object detection is needed by changing the data set.

# Bibliography:

[1] Dang Ha the Hien. "The Modern History of Object Recognition – Infographic". *Medium*, 2017. [Online] Available: https://medium.com/@nikasa1889/the-modern-history-of-object-recognition-infographic-aea18517c318 [Accessed: March 2020]

[2] Prabhu. "Understanding of Convolutional Neural Network (CNN) – Deep Learning". *Medium*, 2018. [Online] Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 [Accessed: March 2020]

[3] Shyamal Patel, Johanna Pingel. "*Introduction to Deep Learning: What Are Convolutional Neural Networks?*". MathWorks, 2017. [Online] Available: https://es.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html [Accessed: March 2020]

[4] Ujjwalkarn. "*An Intuitive Explanation of Convolutional Neural Networks*". The Data Science Blog, 2016. [Online] Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ [Accessed: March 2020]

[5] Joyce Xu. "*Deep Learning for Object Detection: A Comprehensive Review*". Towards Data Science, 2017. [Online] Available: https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9 [Accessed: March 2020]

[6] R. Girshick, J. Donahue, T. Darrell, J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". *IEEE Conference on Computer Vision and Pattern Recognition.* October 2014. California, USA. Pp 580-587.

[7] R. Girshick. "Fast R-CNN". *IEEE Conference on Computer Vision (ICCV)*. September 2015. Santiago, Chile. Pp 1440-1448.

[8] J. Hui. "*Fast R-CNN and Faster R-CNN*". Jonathan Hui Blog, 2017. [Online] Available: https://jhui.github.io/2017/03/15/Fast-R-CNN-and-Faster-R-CNN/ [Accessed: March 2020]

[9] S. Ren, K. He, R. Girshick, J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2015. vol. 39, pp 1137-1149.

[10] OpenCV. "*Operations on arrays – flip"*. Open Source Computer Vision. [Online] Available: https://docs.opencv.org/4.0.0/d2/de8/group__core__array.html#gaca7be533e3dac7feb70fc60635adf441 [Accessed: April 2020]

[11] Pkulzc. "*Faster R-CNN Inception Resnet v2"*. GitHub, 2019. [Online] Available: https://github.com/tensorflow/models/blob/master/research/object_detection/models/faster_rcnn_inception_resnet_v2_feature_extractor.py [Accessed: April 2020]

[12] TensorFlow. "*TensorFlow Object Detection API*". TensorFlow. [Online] Available: https://www.tensorflow.org/lite/models/object_detection/overview [Accessed: April 2020]

[13] V. Bushaev. "*How do we 'train' neural networks?"*. Towards Data Science, 2017. [Online] Available: https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73 [Accessed: April 2020]

[14] M. Rouse. "*RESTful API (REST API)"*. search app Architecture, 2020. [Online] Available: https://searchapparchitecture.techtarget.com/definition/RESTful-API#:~:text=A%20RESTful%20API%20is%20an,to%20communicate%20with%20each%20other [Accessed: May 2020]

[15] M. Rouse. "*REST (REpresentational State Transfer)"*. search app Architecture, 2019. [Online] Available: https://searchapparchitecture.techtarget.com/definition/REST-REpresentational-State-Transfer [Accessed: May 2020]

[16] Billyrrr. "*Easy OpenAPI specs and Swagger UI for your Flask API - Flasgger"*. GitHub, 2019. [Online] Available: https://github.com/flasgger/flasgger [Accessed: May 2020]

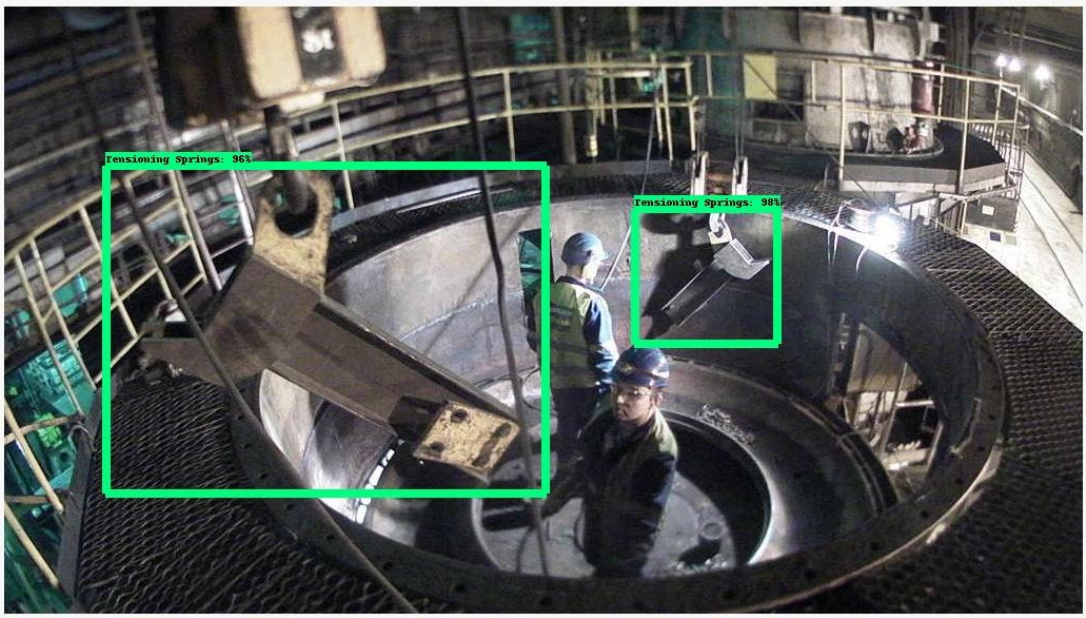
# Appendices:

## A. Example of detected objects in images

Tensioning Spring:

Image status

Thread work with job id 123 has stopped

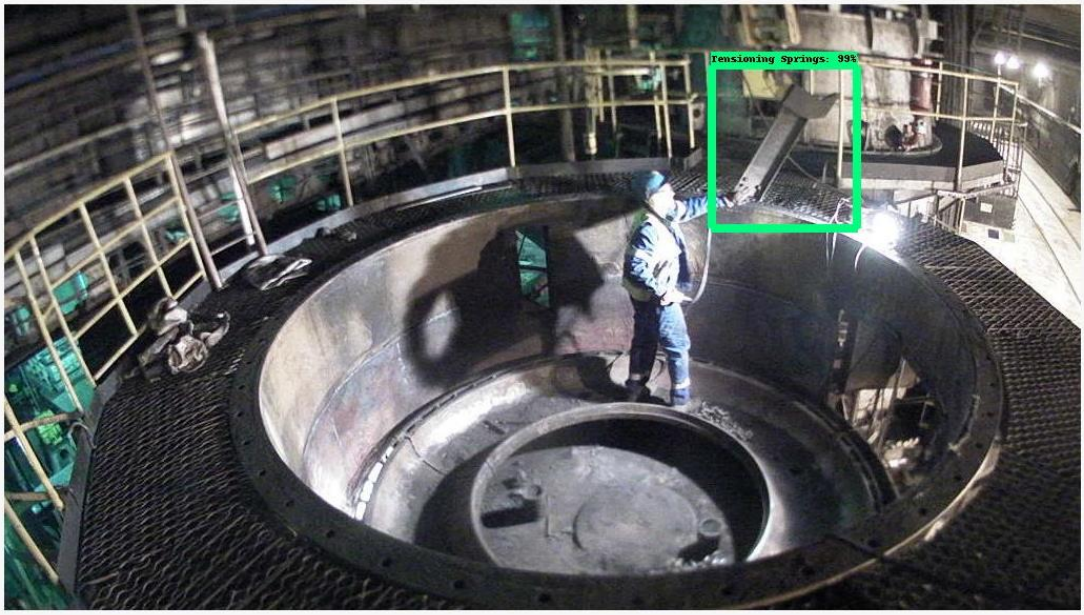




```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Tensioning Springs",
            "certainty": "0.99164283",
            "box": [
                378,
                105,
                456,
                378
            ],
            "properties": {}
        }
    ]
}
```

## Image status

Thread work with job id 123 has stopped





```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Tensioning Springs",
            "certainty": "0.99164283",
            "box": [
                378,
                105,
                456,
                378
            ],
            "properties": {}
        }
    ]
}
```

Pressure Spring:

Image status

Thread work with job id 123 has stopped



```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Pressure Spring",
            "certainty": "0.9899255",
            "box": [
                281,
                136,
                475,
                361
            ],
            "properties": {}
        }
    ]
}
```

## Image status

Thread work with job id 123 has stopped



```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Pressure Spring",
            "certainty": "0.9986883",
            "box": [
                236,
                614,
                431,
                902
            ],
            "properties": {}
        }
    ]
}
```

Metal Sheet:

Image status

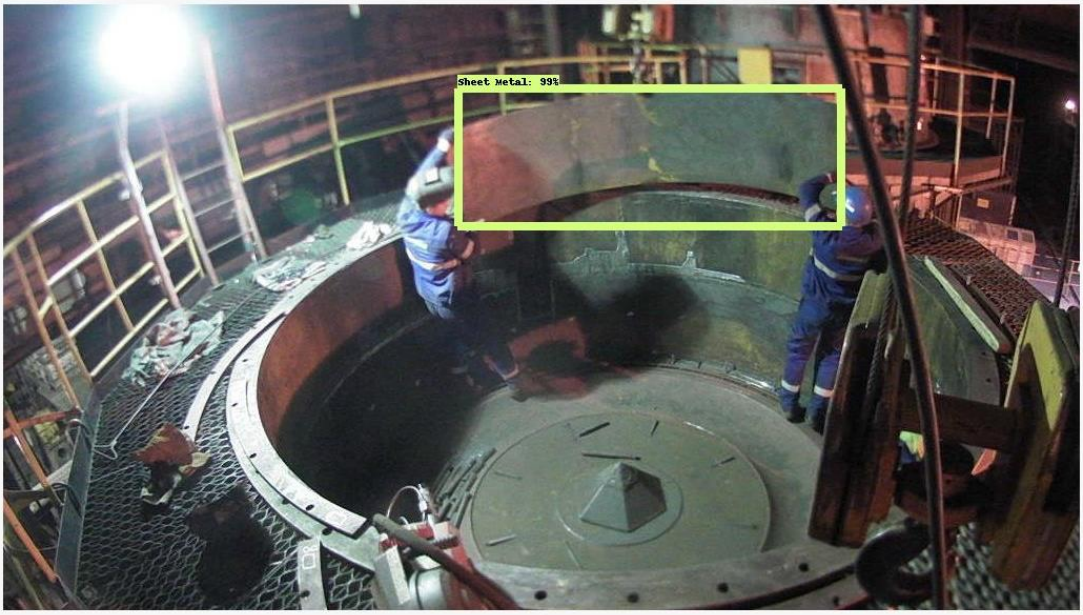
Thread work with job id 123 has stopped



```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Sheet Metal",
            "certainty": "0.99958867",
            "box": [
                347,
                174,
                551,
                419
            ],
            "properties": {}
        }
    ]
}
```

## Image status

Thread work with job id 123 has stopped



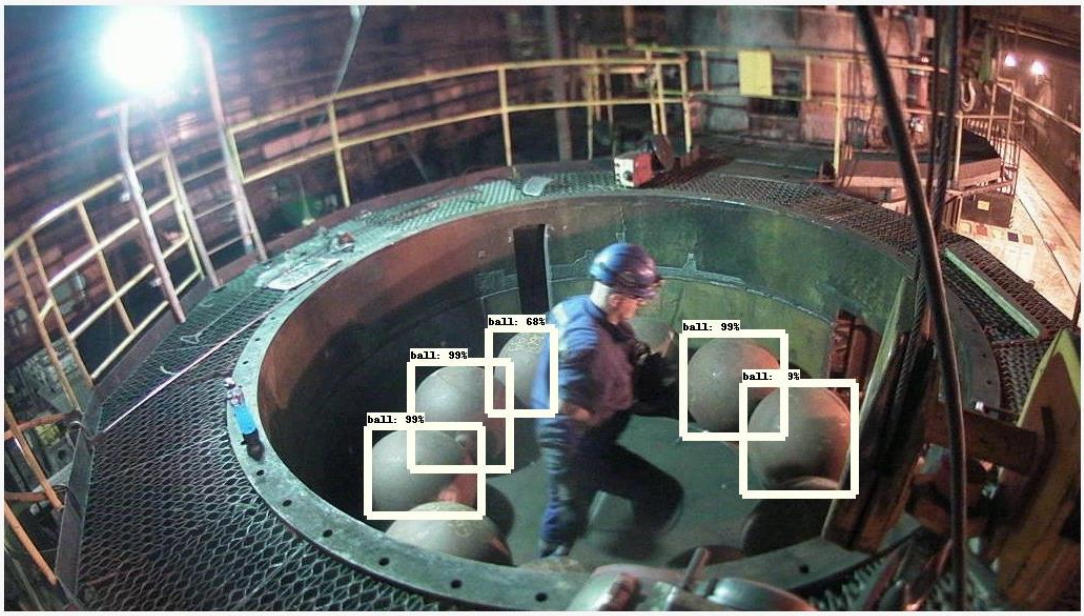


```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "Sheet Metal",
            "certainty": "0.9991929",
            "box": [
                243,
                144,
                449,
                374
            ],
            "properties": {}
        }
    ]
}
```

Balls:

Image status

Thread work with job id 123 has stopped

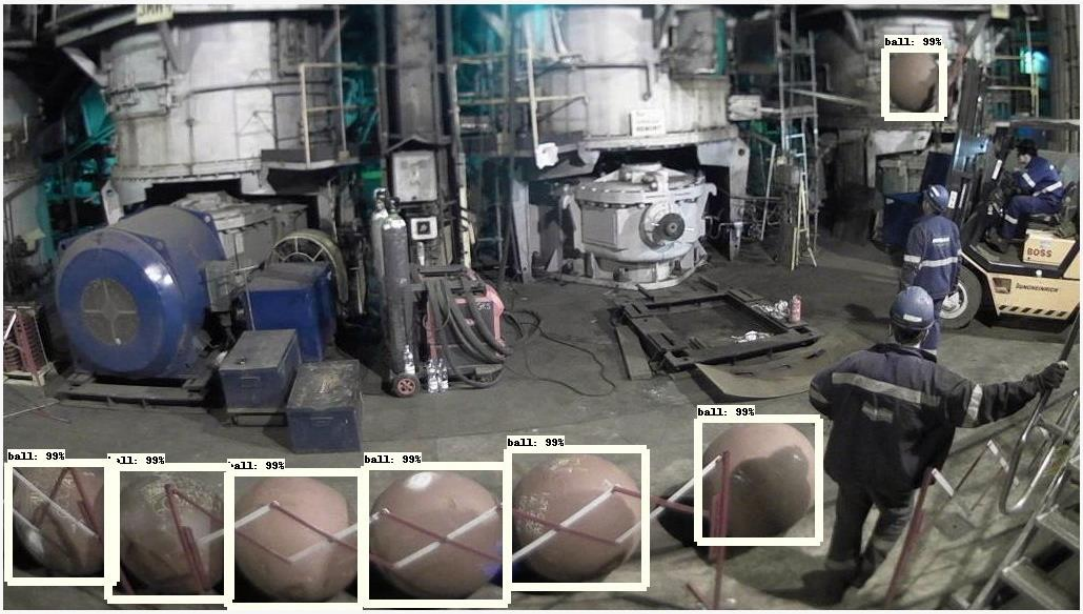




```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "ball",
            "certainty": "0.9998123",
            "box": [
                395,
                638,
                455,
                826
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.99975866",
            "box": [
                218,
                604,
                271,
                784
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.9997353",
            "box": [
                363,
                556,
                418,
                728
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.9979532",
            "box": [
                194,
                712,
                255,
                862
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.68677026",
            "box": [
                259,
                546,
                295,
                689
            ],
            "properties": {}
        }
    ]
}
```

## Image status

Thread work with job id 123 has stopped





```
{
    "result": "OK",
    "objects": [
        {
            "recognized_class": "ball",
            "certainty": "0.99989486",
            "box": [
                193,
                781,
                270,
                1012
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.99988735",
            "box": [
                120,
                791,
                192,
                1017
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.99984765",
            "box": [
                371,
                701,
                437,
                907
            ],
            "properties": {}
        },
```

```
        {
            "recognized_class": "ball",
            "certainty": "0.9998405",
            "box": [
                56,
                784,
                122,
                1006
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.9998154",
            "box": [
                270,
                753,
                344,
                984
            ],
            "properties": {}
        },
        {
            "recognized_class": "ball",
            "certainty": "0.99968135",
            "box": [
                3,
                775,
                60,
                975
            ],
            "properties": {}
        },
```

```
        {
            "recognized_class": "ball",
            "certainty": "0.99964356",
            "box": [
                471,
                76,
                503,
                189
            ],
            "properties": {}
        }
    ]
}
```

# Glossary

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **JSON** | JavaScript Object Notation |
| **API** | Application Programming Interface |
| **REST** | Representational State Transfer |
| **CNN** | Convolutional Neural Network |
| **R-CNN** | Region-based Convolutional Neural Network |
| **SVM** | Support Vector Machine |
| **RPN** | Region Proposal Network |
| **RoI pooling** | Region of Interest pooling |
| **XML** | Extensible Mark-up Language |
| **SOA** | Service-Oriented Architecture |
| **SOAP** | Simple Object Access Protocol |
| **TP** | True Positive |
| **FP** | False Positive |
| **TN** | True Negative |
| **FN** | False Negative |